MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2015/2016

TPL 2141 – PROGRAMMING LANGUAGE CONCEPT

(All sections / Groups)

5 OCT 2015
2.30 p.m – 4.30 p.m
(2 Hours)

INSTRUCTIONS TO STUDENTS

1. This Question paper consists of 4 pages (excluding cover page) with 6 Questions.
2. Attempt **FIVE** out of **SIX** questions. All questions carry equal marks and the distribution of the marks for each question is given.
3. Please write all your answers in the Answer Booklet provided

Question 1

(a) It is useful for a programmer to have some background about the concept and design of multiple programming languages, even though he or she may never actually design a programming language. Discuss **FOUR** potential benefits of learning the concepts of multiple programming languages

[2 marks]

(b) Assume that you are being selected as the principal programmer to develop Windows-based software.

(i) Explain how you choose a programming language based on any **TWO** language evaluation criteria, such as readability, writability, reliability, cost, portability, generality and well-definedness.

(ii) Discuss the general implementation method (with diagram that illustrates the implementation process) of your selected programming language.

(iii) Briefly describe the programming environment (collection of tools, such as text editor and compiler) that you have selected to develop the software.

[2 + 2 + 2 = 6 marks]

(c) List out the **FOUR** main properties of Backus-Naur Form (BNF).

[2 marks]

Question 2

Consider the following grammar with terminal symbols in **bold**:

expression → **identifier** | **number** | - expression
| (expression)
| expression operator expression
operator → + | - | * | /

(a) Show that the following sentence is valid for the grammar by using rightmost derivation (do not draw parse tree): `slope * x + intercept`

[3 marks]

(b) Draw the parse tree based on the derivation in (a).

[2 marks]

(c) Why is the grammar ambiguous? Prove with the sentence in (a).

[3 marks]

(d) Construct a syntax diagram for the first rule: expression.

[2 marks]

Continued...

Question 3

(a) Consider the following BNF.

⟨id⟩ → ⟨letter⟩ | ⟨id⟩ ⟨letter⟩ | ⟨id⟩ ⟨digits⟩
⟨digits⟩ → ⟨digit⟩ | ⟨digit⟩ ⟨digits⟩

(i) Convert the BNF into Extended Backus-Naur Form (EBNF).
(ii) Draw the syntax diagram based on your EBNF in (i).

[2 + 2 = 4 marks]

(b) Consider the following attribute grammar.

Syntax rule 1: <assign> → <var> = <expr>
Semantic rule 1: <expr>.expected_type ← <var>.actual_type

Syntax rule 2: <expr> → <var>[2] + <var>[3]
Semantic rule 2: <expr>.actual_type ← if (<var>[2].actual_type = int) and (<var>[3].actual_type = int) then int else real
Predicate: <expr>.actual_type = <expr>.expected_type

(i) Identify the **synthesized** attribute and **inherited** attribute from the Semantic rule 1 and 2.
(ii) What is the purpose of **predicate** function in the attribute grammar?
(iii) Discuss the main usage of attribute grammar for BNF.

[2 + 2 + 2 = 6 marks]

Question 4

(a) All three of the implementation approaches (compilation, pure interpretation and hybrid implementation) use both lexical and syntax analyzers. Explain **THREE** reasons why lexical analysis is separated from syntax analysis?

[3 marks]

(b) Consider the following grammar for Boolean expressions.

<BExp> → **false** | **true** |
<BExp> **and** <BExp> |
<BExp> **or** <BExp> |
not <BExp>

(i) Why is this grammar not suitable for top-down parsing? Please explain.
(ii) Show the bottom-up parsing with shift-reduce parser for this sentence: `false and true and false`

[2 + 5 = 7 marks]

Continued...

Question 5

(a) There are two types of control statements in C++ programming language. Give **TWO** examples of C++ codes for each control statements (selection and iterative statements).

[4 marks]

(b) For each of the following parameter-passing methods, what are the values of the output for `relax` after execution?

```
void enjoy (int play) {
   play ++;
   relax = relax + 2;
}
int main() {
   int relax = 1;

   enjoy (relax);
   cout << relax << endl;
   return 0;
}
```

(i) passed by value
(ii) passed by reference
(iii) passed by value-result

[1 + 1 + 1 = 3 marks]

(c) C, C++, Python, and C# have unconditional unlabeled exits but Java and Perl have unconditional labeled exits.

(i) What is the unconditional unlabeled exit provided by C program?

(ii) How is it different from the unconditional labeled exit of Java program? Support your opinion with sample Java codes.

[1 + 2 = 3 marks]

Continued...

Question 6

(a) Given the following C program:

```
void fun1(void);  /* prototype */
void fun2(void);  /* prototype */
void fun3(void);  /* prototype */
void main() {
      int a, b, c;
      . . .
}
void fun1(void) {
      int b, c, d;
      . . .
}
void fun2(void) {
      int c, d, e;
      . . .
}
void fun3(void) {
      int d, e, f;
      . . .
}
```

Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last function called? Include with each visible variable the name of the function in which it was defined.

(i) main calls fun1; fun1 calls fun3; fun3 calls fun2

(ii) main calls fun3; fun3 calls fun1

[2 + 2 = 4 marks]

(b) What is the advantage of short-circuit evaluation. How is the short-circuit evaluation happens in the following Boolean expression? Please explain.

```
(a >= 0) && (b < 10)
```

[4 marks]

(c) A mixed-mode expression is an expression that has operands of different types. Type conversion is used for handling the execution of mixed-mode expression. Discuss the TWO ways of type conversion with sample codes.

[2 marks]

End of Paper.